

Standard Code Library

FLself

SCUT

August 10, 2023

Contents

一切的开始

一些宏定义

- 需要 C++11

```
1 // #define DEBUG
2 // #define InTerminal
3 // #define std_cpp17
4 #include<bits/stdc++.h>
5 #define int long long
6 #define PII std::pair<int, int>
7 #define VI std::vector<int>
8 #define VPII std::vector<std::pair<int, int> >
9 #define VVI std::vector<std::vector<int> >
10 #define ALL(a) (a).begin(), (a).end()
11 #define SIZ(a) ((int)(a).size())
12 #define FOR(i, l, r) for (int i = (l); i <= (r); ++i)
13 #define REP(i, r, l) for (int i = (r); i >= (l); --i)
14 #define lowbit(x) ((x) & -(x))
15 #define lbpos(x) (__builtin_ctz(x))
16 #define hbpos(x) (31 - __builtin_clz(x))
17
18 template<typename S, typename T> std::istream &operator>>(std::istream &is, std::pair<S, T> &pp) { is >> pp.first >>
    ⇨ pp.second; return is; }
19 template<typename S, typename T> std::ostream &operator<<(std::ostream &os, std::pair<S, T> pp) { os << "(" <<
    ⇨ pp.first << ", " << pp.second << ")"; return os; }
20 template<typename S, std::size_t _siz> std::istream &operator>>(std::istream &is, std::array<S, _siz> &arr) { for
    ⇨ (auto &x: arr) is >> x; return is; }
21 template<typename S, std::size_t _siz> std::ostream &operator<<(std::ostream &os, std::array<S, _siz> arr) { os <<
    ⇨ "("; for (auto x: arr) os << x << ", "; os << ")"; return os; }
22 template<typename T> std::istream &operator>>(std::istream &is, std::vector<T> &vec) { for (auto &x: vec) is >> x;
    ⇨ return is; }
23 template<typename T> std::ostream &operator<<(std::ostream &os, const std::vector<T> &vec) { os << '{'; for (auto
    ⇨ &x: vec) os << x << ", "; return os << "}"; }
24 #ifndef std_cpp17
25 template<class Tuple, std::size_t... Is> void print_tuple_impl(std::ostream &os, const Tuple &t,
    ⇨ std::index_sequence<Is...>) { ((os << (Is == 0? "" : ", ") << std::get<Is>(t), ...); }
26 template<class... Args> std::ostream &operator<<(std::ostream &os, const std::tuple<Args...> &t) { os << "(";
    ⇨ print_tuple_impl(os, t, std::index_sequence_for<Args...>{}); return os << ")"; }
27 #endif
28 #ifndef DEBUG
29 #ifndef InTerminal
30 #define dbg(x...) do { std::cerr << "\033[32;1m" << #x << " -> "; err(x); } while (0)
31 void err() { std::cerr << "\033[39;0m" << std::endl; }
32 #else
33 #define dbg(x...) do { std::cerr << #x << " -> "; err(x); } while (0)
34 void err() { std::cerr << std::endl; }
35 #endif
36 template<typename T, typename... A>
37 void err(T a, A... x) { std::cerr << a << ' '; err(x...); }
38 #else
39 #define dbg(...)
40 #endif
41
42 using namespace std;
43 const int maxn = 2e5 + 3;
44 const int INF = 0x3f3f3f3f3f3f3f3f;
45 const int mod = 998244353;
46 mt19937 RD(time(0));
47
48
49 void solv() {
50
51     return ;
52 }
53
54
55 signed main() {
56     // freopen("./data.in", "r", stdin);
57     std::ios::sync_with_stdio(false), std::cin.tie(0), std::cout.tie(0);
```

```

58     int beg__TT = clock();
59
60     signed _ttt;
61     cin >> _ttt;
62
63     while(_ttt--)
64         solv();
65
66     #ifdef DEBUG
67     std::cerr << "use : " << (clock() - beg__TT) << "ms\n";
68     #endif
69     return 0;
70 }

```

数据结构

ST 表

- 一维

```

1  class Sparcetable {
2      vector<vector<int>> > st;
3      int siz;
4      bool MX_flg = 0;
5      inline int renew(int x, int y) {
6          if (MX_flg) return max(x, y);
7          return min(x, y);
8      }
9  public:
10     // 注意 hbpos(0) 返回-1
11     bool (*comp)(int, int);
12     Sparcetable():siz(maxn) {st.resize(hbpos(maxn) + 1, std::vector<int> (maxn));}
13     Sparcetable(const std::vector<int>& a, bool _MX_flg = 1): siz(a.size()), MX_flg(_MX_flg) {
14         int n = a.size();
15         st.resize(hbpos(n) + 1, vector<int> (n + 1));
16         for (int i = 1; i <= n; ++i) st[0][i] = a[i];
17         for (int i = 1; i <= hbpos(siz); ++i) {
18             for (int j = 1; j + (1 << i) <= siz + 1; ++j) {
19                 st[i][j] = renew(st[i - 1][j], st[i - 1][j + (1 << (i - 1))]);
20             }
21         }
22     }
23     int query(int l, int r) {
24         int len = hbpos(r - l + 1);
25         return renew(st[len][l], st[len][r - (1 << len) + 1]);
26     }
27 };
28

```

- 二维

```

1  int f[10][10][maxn][maxn];
2  #define _highbit(x) (31 - __builtin_clz(x))
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[p][q][x][y], f[p][q][xx - (1 << p) + 1][yy - (1 << q) + 1]),
6          max(f[p][q][xx - (1 << p) + 1][y], f[p][q][x][yy - (1 << q) + 1])
7      );
8  }
9  void init() {
10     for (int x = 0; x <= _highbit(n); ++x)
11     for (int y = 0; y <= _highbit(m); ++y)
12     for (int i = 0; i <= n - (1 << x); ++i)
13     for (int j = 0; j <= m - (1 << y); ++j) {
14         if (!x && !y) { f[x][y][i][j] = a[i][j]; continue; }
15         f[x][y][i][j] = calc(
16             i, j,
17             i + (1 << x) - 1, j + (1 << y) - 1,
18             max(x - 1, 0), max(y - 1, 0)
19         );

```

```

20     }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, _highbit(xx - x + 1), _highbit(yy - y + 1));
24 }

```

Fenwick Tree(树状数组)

● 一维

```

1  template<typename T>
2  class FenwickT {
3      int n;
4      vector<T> tr;
5  public:
6      FenwickT(int siz): tr(siz), n(siz) {}
7      FenwickT(int siz, T ini): tr(siz, ini), n(siz) {}
8      void add(int p, T x) {
9          for (int i = p; i < n; i += i & (-i)) tr[i] += x;
10     }
11     T query(int p) {
12         T ret = T(0);
13         for (int i = p; i > 0; i -= i & (-i)) ret += tr[i];
14         return ret;
15     }
16     T range_sum(int l, int r) {
17         return (query(r) - query(l - 1));
18     }
19     int kth(T k) {
20         int x = 0;
21         for (int i = 1 << std::lg(n); i; i /= 2) {
22             if (x + i <= n && k >= a[x + i - 1]) {
23                 x += i;
24                 k -= a[x - 1];
25             }
26         }
27         return x;
28     }
29 };

```

并查集

```

1  class DSU {
2      std::vector<int> fa, siz;
3  public:
4      DSU(int n): fa(n), siz(n, 1) {std::iota(fa.begin(), fa.end(), 0);}
5      int findfa(int x) {
6          int rt = x;
7          while (rt != fa[rt]) rt = fa[rt];
8          for (int i = x; i != rt; i = x) {
9              siz[i] -= siz[x];
10             x = fa[i], fa[i] = rt;
11         }
12         return rt;
13     }
14     bool merge(int x, int y) {
15         x = findfa(x), y = findfa(y);
16         if (x == y) return false;
17         if (siz[x] > siz[y])
18             fa[y] = x, siz[x] += siz[y];
19         else
20             fa[x] = y, siz[y] += siz[x];
21         return true;
22     }
23     bool same(int x, int y) { return findfa(x) == findfa(y); }
24     int size(int x) { return siz[findfa(x)]; }
25 };

```

带权并查集

```
1 class DSU {
2     std::vector<int> fa, siz;
3     std::vector<int> tag;
4 public:
5     DSU(int n): fa(n), siz(n, 1), tag(n, 0) {std::iota(fa.begin(), fa.end(), 0);}
6     int findfa(int x) {
7         int rt = x, tg = tag[x];
8         while (rt != fa[rt]) rt = fa[rt], tg ^= tag[rt];
9         for (int i = x; i != rt; i = x) {
10             siz[i] -= siz[x];
11             x = fa[i], fa[i] = rt;
12             // std::swap(tg, tag[i]); tg ^= tag[i];
13             tg ^= tag[i] ^= tg ^= tag[i] ^= tg;
14         }
15         return rt;
16     }
17     bool merge(int x, int y, int tg = 0) {
18         if (findfa(x) == findfa(y)) {
19             if ((tag[x] ^ tag[y]) != tg) return false;
20             return true;
21         }
22         tg ^= tag[x] ^ tag[y];
23         x = findfa(x), y = findfa(y);
24         if (siz[x] > siz[y])
25             fa[y] = x, siz[x] += siz[y], tag[y] = tg;
26         else
27             fa[x] = y, siz[y] += siz[x], tag[x] = tg;
28         return true;
29     }
30     bool same(int x, int y) { return findfa(x) == findfa(y); }
31     int size(int x) { return siz[findfa(x)]; }
32 };
33
```

线段树

```
1 class SegT{
2     struct Pt{
3         int val, ls, rs, sl;
4         Pt():val(0), ls(0), rs(0), sl(0) {}
5         Pt(int v, int l, int r, int s): val(v), ls(l), rs(r), sl(s) {}
6         ~Pt() {}
7     };
8     int renew(int x, int y) {
9         return x + y;
10    }
11    int renewsl(int x, int y) {
12        return x + y;
13    }
14    int renewqj(int x, int y, int l, int r) {
15        return x + y * (r - l + 1);
16    }
17 public:
18     std::vector<Pt> tr;
19     int cnt = 0; // 结点数
20     SegT() {cnt = 1; tr.emplace_back();} // 建一棵空树
21
22     // 修改
23     void add(int l, int r, int p, int ql, int qr, int x) {
24         if (ql <= l && r <= qr) {
25             tr[p].val = renewqj(tr[p].val, x, l, r);
26             tr[p].sl = renewsl(tr[p].sl, x);
27             return;
28         }
29         int mid = l + (r - l) / 2;
30         if (tr[p].sl) {
31             if (!tr[p].ls) tr[p].ls = cnt++;
32             if (!tr[p].rs) tr[p].rs = cnt++;
33             if (cnt >= tr.size()) tr.resize(cnt + 1);

```

```

34         int ls = tr[p].ls, rs = tr[p].rs;
35         tr[ls].val = renewqj(tr[ls].val, tr[p].sl, l, mid), tr[rs].val = renewqj(tr[rs].val, tr[p].sl, mid + 1, r);
36         tr[ls].sl = renewsl(tr[ls].sl, tr[p].sl), tr[rs].sl = renewsl(tr[rs].sl, tr[p].sl);
37         tr[p].sl = 0;
38     }
39     if (ql <= mid) {
40         if (!tr[p].ls) tr[p].ls = cnt++;
41         if (cnt >= tr.size()) tr.resize(cnt + 1);
42         add(l, mid, tr[p].ls, ql, qr, x);
43     }
44     if (qr > mid) {
45         if (!tr[p].rs) tr[p].rs = cnt++;
46         if (cnt >= tr.size()) tr.resize(cnt + 1);
47         add(mid + 1, r, tr[p].rs, ql, qr, x);
48     }
49     tr[p].val = renew(tr[p].ls? tr[tr[p].ls].val: 0, tr[p].rs? tr[tr[p].rs].val: 0);
50 };
51
52 // 查询
53 int query(int l, int r, int p, int ql, int qr) {
54     if (ql <= l && r <= qr) return tr[p].val;
55     int mid = l + (r - l) / 2;
56     if (tr[p].sl) {
57         if (!tr[p].ls) tr[p].ls = cnt++;
58         if (!tr[p].rs) tr[p].rs = cnt++;
59         if (cnt >= tr.size()) tr.resize(cnt + 1);
60         int ls = tr[p].ls, rs = tr[p].rs;
61         tr[ls].val = renewqj(tr[ls].val, tr[p].sl, l, mid), tr[rs].val = renewqj(tr[rs].val, tr[p].sl, mid + 1, r);
62         tr[ls].sl = renewsl(tr[ls].sl, tr[p].sl), tr[rs].sl = renewsl(tr[rs].sl, tr[p].sl);
63         tr[p].sl = 0;
64     }
65     int ret = 0;
66     if (ql <= mid) {
67         if (!tr[p].ls) tr[p].ls = cnt++;
68         if (cnt >= tr.size()) tr.resize(cnt + 1);
69         ret = renew(ret, query(l, mid, tr[p].ls, ql, qr));
70     }
71     if (qr > mid) {
72         if (!tr[p].rs) tr[p].rs = cnt++;
73         if (cnt >= tr.size()) tr.resize(cnt + 1);
74         ret = renew(ret, query(mid + 1, r, tr[p].rs, ql, qr));
75     }
76     tr[p].val = renew(tr[p].ls? tr[tr[p].ls].val: 0, tr[p].rs? tr[tr[p].rs].val: 0);
77     return ret;
78 };
79 };
80
81 // 静态线段树
82 template<class Info,
83         class Merge = std::plus<Info> >
84 struct SegmentTree {
85     const int n;
86     const Merge merge;
87     std::vector<Info> info;
88     SegmentTree(int n) : n(n), merge(Merge()), info(4 << std::lg(n)) {}
89     SegmentTree(std::vector<Info> init) : SegmentTree(init.size()) {
90         std::function<void(int, int, int)> build = [&](int p, int l, int r) {
91             if (r - l == 1) {
92                 info[p] = init[l];
93                 return;
94             }
95             int m = (l + r) / 2;
96             build(2 * p, l, m);
97             build(2 * p + 1, m, r);
98             pull(p);
99         };
100         build(1, 0, n);
101     }
102     void pull(int p) {
103         info[p] = merge(info[2 * p], info[2 * p + 1]);
104     }

```

```

105 void modify(int p, int l, int r, int x, const Info &v) {
106     if (r - l == 1) {
107         info[p] = v;
108         return;
109     }
110     int m = (l + r) / 2;
111     if (x < m) {
112         modify(2 * p, l, m, x, v);
113     } else {
114         modify(2 * p + 1, m, r, x, v);
115     }
116     pull(p);
117 }
118 void modify(int p, const Info &v) {
119     modify(1, 0, n, p, v);
120 }
121 Info rangeQuery(int p, int l, int r, int x, int y) {
122     if (l >= y || r <= x) {
123         return Info();
124     }
125     if (l >= x && r <= y) {
126         return info[p];
127     }
128     int m = (l + r) / 2;
129     return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
130 }
131 Info rangeQuery(int l, int r) {
132     return rangeQuery(1, 0, n, l, r);
133 }
134 };

```

吉如一线段树

区间赋值、区间取 min, 求和、求最大值、求历史最大值

```

1 namespace JRY_SegT{
2     // namespace JRY_SegT
3     // O(n log n) luogu n,m 5e5 max1.62s
4     using i64 = long long;
5     constexpr i64 INF = 0x7fffffffffffffff;
6     struct Info {
7         i64 sum,mx,smx,hmx,cnt;
8         Info(i64 sum=0,i64 mx=-INF,i64 smx=-INF,i64 hmx=-INF,i64 cnt=0)
9             : sum(sum),mx(mx),smx(smx),hmx(hmx),cnt(cnt) {}
10    };
11    Info merge(Info x,Info y) {
12        Info r;
13        r.sum = x.sum + y.sum;
14        r.mx = std::max(x.mx, y.mx);
15        r.hmx = std::max(x.hmx, y.hmx);
16        if(x.mx == y.mx) r.smx = std::max(x.smx, y.smx), r.cnt = x.cnt + y.cnt;
17        else if(x.mx > y.mx) r.smx = std::max(x.smx, y.mx), r.cnt = x.cnt;
18        else r.smx = std::max(x.mx, y.smx), r.cnt = y.cnt;
19        return r;
20    }
21    struct Tag {
22        i64 lzm,lzo,hlzm,hlzo;
23        Tag() : lzm(0),lzo(0),hlzm(0),hlzo(0) {}
24        void clear() {
25            lzm = lzo = hlzm = hlzo = 0;
26        }
27        bool is0() {
28            return lzm == 0 && lzo == 0 && hlzm == 0 && hlzo == 0;
29        }
30    };
31    struct SegT {
32        int n;
33        std::vector<Info> t;
34        std::vector<Tag> z;
35        std::vector<int> le,ri;
36        SegT(int n,const std::vector<i64>&a) : n(n), t(4*n), z(4*n), le(4*n), ri(4*n) {

```



```

37     build(1,0,n,a);
38 }
39 void build(int p,int l,int r,const std::vector<i64>&a) {
40     le[p] = l, ri[p] = r;
41     if(r - l == 1) {
42         t[p] = Info(a[l],a[l],-INF,a[l],1);
43         return;
44     }
45     int mid = l + r >> 1;
46     build(p << 1, l, mid, a);
47     build(p << 1 | 1, mid, r, a);
48     pull(p);
49 }
50 void pull(int p) {
51     t[p] = merge(t[p << 1], t[p << 1 | 1]);
52 }
53 void update(int p,i64 lzm,i64 lzo,i64 hlzm,i64 hlzo) {
54     t[p].sum += lzm * t[p].cnt + (ri[p] - le[p] - t[p].cnt) * lzo;
55     t[p].hmx = std::max(t[p].hmx, t[p].mx + hlzm);
56     t[p].mx += lzm;
57     if(t[p].smx != -INF) t[p].smx += lzo;
58     z[p].hlzm = std::max(z[p].hlzm, z[p].lzm + hlzm), z[p].lzm += lzm;
59     z[p].hlzo = std::max(z[p].hlzo, z[p].lzo + hlzo), z[p].lzo += lzo;
60 }
61 void pushdown(int p) {
62     if(z[p].is0()) return;
63     int mx = std::max(t[p << 1].mx, t[p << 1 | 1].mx);
64     for(auto ps : {p << 1, p << 1 | 1}) {
65         if(t[ps].mx == mx) update(ps, z[p].lzm, z[p].lzo, z[p].hlzm, z[p].hlzo);
66         else update(ps, z[p].lzo, z[p].lzo, z[p].hlzo, z[p].hlzo);
67     }
68     z[p].clear();
69 }
70 void range_add(int p,int L,int R,i64 v) {
71     if(ri[p] <= L || R <= le[p]) return;
72     if(L <= le[p] && ri[p] <= R) {
73         update(p,v,v,v,v);
74         return;
75     }
76     pushdown(p);
77     range_add(p << 1, L, R, v);
78     range_add(p << 1 | 1, L, R, v);
79     pull(p);
80 }
81 void range_min(int p,int L,int R,i64 v) {
82     if(ri[p] <= L || R <= le[p] || t[p].mx <= v) return;
83     if(L <= le[p] && ri[p] <= R && t[p].smx < v) {
84         update(p, v - t[p].mx, 0, v - t[p].mx, 0);
85         return;
86     }
87     pushdown(p);
88     range_min(p << 1, L, R, v);
89     range_min(p << 1 | 1, L, R, v);
90     pull(p);
91 }
92 i64 query_sum(int p,int L,int R) {
93     if(ri[p] <= L || R <= le[p]) return 0ll;
94     if(L <= le[p] && ri[p] <= R) return t[p].sum;
95     pushdown(p);
96     return query_sum(p << 1, L, R) + query_sum(p << 1 | 1, L, R);
97 }
98 i64 query_mx(int p,int L,int R) {
99     if(ri[p] <= L || R <= le[p]) return -INF;
100    if(L <= le[p] && ri[p] <= R) return t[p].mx;
101    pushdown(p);
102    return std::max(query_mx(p << 1, L, R), query_mx(p << 1 | 1, L, R));
103 }
104 i64 query_hmx(int p,int L,int R) {
105     if(ri[p] <= L || R <= le[p]) return -INF;
106     if(L <= le[p] && ri[p] <= R) return t[p].hmx;
107     pushdown(p);

```

```

108         return std::max(query_hmx(p << 1, L, R), query_hmx(p << 1 | 1, L, R));
109     }
110 };
111 };
112

```

主席树

可持久化线段树, 感觉主要是利用单点 \log 次的性质以及离散化的思路. 比如区间第 k 小就是构造 n 个线段树 (每多加一个实际上只多加了 \log 个点), 然后再用离散化的点 (实际上就是其排名值) 找到前缀和之差为 k (此时为第 k 小, 可以这样想: 第 r 棵树的 r 个点中包含了第 $l-1$ 棵树的 $l-1$ 个点, 减掉那些点, 剩下的就是区间 $[l, r]$ 的点, 然后找第 k 个) 的点. 用离线不那么方便. 代码暂时先留空了...

```

1  // 主席树
2  // 对权值建立可持久化线段树, 再二分一样求区间第  $k$  大
3  struct HJTtr {
4      std::vector<int> ls, rs, rt;
5      std::vector<int> ind, sum;
6      int tot, _n;
7      int getid(const int& val) {
8          return std::lower_bound(ind.begin(), ind.end(), val) - ind.begin();
9      }
10     int build(int l, int r) {
11         int root = tot++;
12         sum[root] = 0;
13         if (l == r) return root;
14         int mid = l + (r - l) / 2;
15         ls[root] = build(l, mid);
16         rs[root] = build(mid + 1, r);
17         return root;
18     }
19     int update(int k, int x = 1) {
20         auto upd=[&](auto upd, int l, int r, int root) {
21             int dir = tot++;
22             ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + x;
23             if (l == r) return dir;
24             int mid = l + (r - l) / 2;
25             if (k <= mid) ls[dir] = upd(upd, l, mid, ls[dir]);
26             else rs[dir] = upd(upd, mid + 1, r, rs[dir]);
27             return dir;
28         };
29         return upd(upd, 0, _n - 1, rt.back());
30     }
31     HJTtr(int n): ls(n * 20), rs(n * 20), ind(n), sum(n * 20), tot(0), _n(n) {}
32     void init(const std::vector<int> &a) {
33         tot = 0;
34         ind.assign(a.begin(), a.end());
35         std::sort(ind.begin(), ind.end());
36         ind.erase(std::unique(ind.begin(), ind.end()), ind.end());
37         rt.push_back(build(0, _n - 1));
38         for (int i = 0; i < a.size(); ++i) rt.push_back(update(getid(a[i])));
39     }
40     /// @brief 查询区间第  $k$  大, 注意区间是左闭右闭。
41     /// @param l 左区间
42     /// @param r 右区间
43     /// @param k 第  $k$  大
44     /// @return
45     /// @note  $l, r, k$  are all 1-indexed.
46     int query(int l, int r, int k) {
47         auto qry=[&](auto qry, int u, int v, int vl, int vr, int kk) {
48             if (vl == vr) return ind[vl];
49             int mid = vl + (vr - vl) / 2, cnt = sum[ls[v]] - sum[ls[u]];
50             if (kk <= cnt) return qry(qry, ls[u], ls[v], vl, mid, kk);
51             else return qry(qry, rs[u], rs[v], mid + 1, vr, kk - cnt);
52         };
53         return qry(qry, rt[l-1], rt[r], 0, _n - 1, k);
54     }
55 };
56
57

```

Trie

```
1  template<typename T>
2  struct Trie {
3      struct Node {
4          T val;
5          int to[2];
6          Node(): val{0} {memset(to, 0, sizeof(to));}
7          Node(T v): val(v) {memset(to, 0, sizeof(to));}
8      };
9      std::vector<Node> a;
10     int cnt;
11     Trie(): a(1), cnt(1) {}
12     Trie(int siz): a(siz), cnt(1) {}
13     int at(char ch) {return ch - '0';}
14     Node& operator [](int idx) {return a[idx];}
15     int go(int cur, char ch) {
16         int &stat = a[cur].to[at(ch)];
17         if (stat == 0) {
18             stat = cnt++;
19             while (cnt > a.size()) a.emplace_back();
20         }
21         return a[cur].to[at(ch)];
22     }
23 };
24
```

笛卡尔树

本质就是一个单调栈，可以用来求一个序列的最大子序列的左右端点

```
1  std::vector<std::array<int, 2> > g;
2  int dikaer(std::vector<int> a) {
3      std::stack<int> stk;
4      g.resize(a.size());
5      for (int i = 1; i < a.size(); ++i) {
6          int las = -1;
7          while (!stk.empty() && a[stk.top()] <= a[i]) las = stk.top(), stk.pop();
8          if (~las) g[i][0] = las;
9          if (!stk.empty()) g[stk.top()][1] = i;
10         stk.push(i);
11     }
12     int rt = -1;
13     while (!stk.empty()) rt = stk.top(), stk.pop();
14     return rt;
15 }
16
```

fhq-treap

```
1  // Definition
2  class Treap {
3  private:
4      struct node {
5          node *left, *right;
6          int size, val, key;
7
8          node();
9          node(int);
10         ~node();
11
12         void pushup();
13     } * root;
14
15     int getNodeSize(node *);
16     node *find(node *, int);
17     std::pair<node *, node *> split(node *, int);
18     std::pair<node *, node *> splitByValue(node *, int);
19     node *merge(node *, node *);
20
```

```

21     public:
22         Treap();
23         ~Treap();
24
25         void insert(int);
26         void erase(int);
27         int getRank(int);
28         int getKth(int);
29         void print(node *p, int dep);
30         void print() { print(root, 0); }
31     };
32
33     // === Treap ===
34
35     // struct Treap::node
36
37     std::mt19937 Rnd(std::chrono::steady_clock::now().time_since_epoch().count());
38
39     Treap::node::node()
40         : left(nullptr), right(nullptr), size(0), val(0), key(Rnd()) {}
41
42     Treap::node::node(int _val)
43         : left(nullptr), right(nullptr), size(1), val(_val), key(Rnd()) {}
44
45     Treap::node::~~node() {
46         delete left, right;
47     }
48
49     inline void Treap::node::pushup() {
50         size = 1;
51         if (left != nullptr) size += left->size;
52         if (right != nullptr) size += right->size;
53     }
54
55     // class Treap
56
57     Treap::Treap()
58         : root(nullptr) {}
59
60     Treap::~~Treap() {
61         delete root;
62     }
63
64     inline int Treap::getNodeSize(Treap::node *node) {
65         return node == nullptr ? 0 : node->size;
66     }
67
68     std::pair<Treap::node *, Treap::node *> Treap::split(Treap::node *p, int k) {
69         if (p == nullptr) return std::make_pair(nullptr, nullptr);
70         std::pair<Treap::node *, Treap::node *> o;
71         if (k <= getNodeSize(p->left)) {
72             o = split(p->left, k);
73             p->left = o.second;
74             p->pushup();
75             o.second = p;
76         } else {
77             o = split(p->right, k - getNodeSize(p->left) - 1);
78             p->right = o.first;
79             p->pushup();
80             o.first = p;
81         }
82         return o;
83     }
84
85     std::pair<Treap::node *, Treap::node *> Treap::splitByValue(Treap::node *p, int val) {
86         if (p == nullptr) return std::make_pair(nullptr, nullptr);
87         std::pair<Treap::node *, Treap::node *> o;
88         if (p->val < val) {
89             o = splitByValue(p->right, val);
90             p->right = o.first;
91             p->pushup();

```

```

92     o.first = p;
93 } else {
94     o = splitByValue(p->left, val);
95     p->left = o.second;
96     p->pushup();
97     o.second = p;
98 }
99 return o;
100 }
101
102 Treap::node *Treap::merge(Treap::node *x, Treap::node *y) {
103     if (x == nullptr) return y;
104     if (y == nullptr) return x;
105     if (x->key > y->key) {
106         x->right = merge(x->right, y);
107         x->pushup();
108         return x;
109     }
110     y->left = merge(x, y->left);
111     y->pushup();
112     return y;
113 }
114
115 Treap::node *Treap::find(Treap::node *p, int val) {
116     if (p == nullptr) return nullptr;
117     if (p->val == val) return p;
118     if (p->val > val) return find(p->left, val);
119     return find(p->right, val);
120 }
121
122 void Treap::insert(int val) {
123     auto o = splitByValue(root, val);
124     o.first = merge(o.first, new Treap::node(val));
125     root = merge(o.first, o.second);
126 }
127
128 void Treap::erase(int val) {
129     auto o = splitByValue(root, val);
130     auto t = o;
131     if (find(o.second, val) != nullptr) {
132         t = split(o.second, 1);
133         delete t.first;
134     }
135     root = merge(o.first, t.second);
136 }
137
138 int Treap::getRank(int val) {
139     auto x = splitByValue(root, val);
140     int r = getNodeSize(x.first) + 1;
141     root = merge(x.first, x.second);
142     return r;
143 }
144
145 int Treap::getKth(int k) {
146     auto x = split(root, k - 1);
147     auto y = split(x.second, 1);
148     Treap::node *o = y.first;
149     root = merge(x.first, merge(y.first, y.second));
150     return o == nullptr ? 0 : o->val;
151 }
152
153 void Treap::print(Treap::node *p, int dep) {
154     if (p == nullptr) return;
155     print(p->left, dep + 1);
156     for (int i = 0; i < dep; ++i) std::cout << " ";
157     std::cout << p->val << '\n';
158     print(p->right, dep + 1);
159 }
160

```

数学

数论

数论整数

```
1  constexpr int P = 998244353;
2  // assume  $-P \leq x < 2P$ 
3  int norm(int x) {
4      //  $x \% P$ ;
5      if (x < 0) { x += P; }
6      if (x >= P) { x -= P; }
7      return x;
8  }
9  template<typename E>
10 E power(E n, int k) {
11     E ret = E(1);
12     while (k) {
13         if (k & 1) ret *= n;
14         n *= n;
15         k >>= 1;
16     } return ret;
17 }
18 struct Z {
19     int x;
20     Z(int x = 0) : x(norm(x)) {}
21     int val() const { return x; }
22     Z operator-() const { return Z(norm(P - x)); }
23     Z inv() const { assert(x != 0); return power(*this, P - 2); }
24     Z &operator*=(const Z &rhs) { x = (long long)(x) * rhs.x % P; return *this; }
25     Z &operator+=(const Z &rhs) { x = norm(x + rhs.x); return *this; }
26     Z &operator-=(const Z &rhs) { x = norm(x - rhs.x); return *this; }
27     Z &operator/=(const Z &rhs) { return *this *= rhs.inv(); }
28     friend Z operator*(const Z &lhs, const Z &rhs) { Z res = lhs; res *= rhs; return res; }
29     friend Z operator+(const Z &lhs, const Z &rhs) { Z res = lhs; res += rhs; return res; }
30     friend Z operator-(const Z &lhs, const Z &rhs) { Z res = lhs; res -= rhs; return res; }
31     friend Z operator/(const Z &lhs, const Z &rhs) { Z res = lhs; res /= rhs; return res; }
32     friend std::istream &operator>>(std::istream &is, Z &a) { long long v; is >> v; a = Z(v); return is; }
33     friend std::ostream &operator<<(std::ostream &os, const Z &a) { return os << a.val(); }
34 };
35 Z operator"" _z(unsigned long long x) { return Z(x); }
```

拉格朗日差值法

```
1  template<typename T>
2  T lintp(const vector<int>& x, const vector<T>& y, int k) {
3      T ans = 0;
4      for (int i = 0; i < x.size(); ++i) {
5          if (k == x[i])
6              return y[i];
7          T u = 1, v = 1;
8          for (int j = 0; j < x.size(); ++j) {
9              if (i == j) continue;
10             u *= (k - x[j]);
11             v *= (x[i] - x[j]);
12         }
13         ans += y[i] * u / v;
14     }
15     return ans;
16 }
```

欧几里得

- 扩展欧几里得

```
1  int exgcd(int a, int b, int& x, int& y) {
2      if (a == 0) {
3          x = 0, y = 1;
4          return b;
5      }
6  }
```

```

6     int ret = exgcd(b % a, a, x, y), xx = x;
7     x = y - b / a * x;
8     y = xx;
9     return ret;
10 }

```

类欧几里得

- $m = \lfloor \frac{a+b}{c} \rfloor$.
- $f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $f(a, b, c, n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a, b, c, n) = nm - f(c, c-b-1, a, m-1)$ 。
- $g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $g(a, b, c, n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a, b, c, n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$ 。
- $h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时, $h(a, b, c, n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2 (n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$; 否则 $h(a, b, c, n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a, b, c, n)$ 。

```

1 template<typename T>
2 T lintp(const vector<int>& x, const vector<T>& y, int k) {
3     T ans = 0;
4     for (int i = 0; i < x.size(); ++i) {
5         if (k == x[i])
6             return y[i];
7         T u = 1, v = 1;
8         for (int j = 0; j < x.size(); ++j) {
9             if (i == j) continue;
10            u *= (k - x[j]);
11            v *= (x[i] - x[j]);
12        }
13        ans += y[i] * u / v;
14    }
15    return ans;
16 }

```

Miller-Rabin 和 Pollard-Rho

- Miller-Rabin

```

1 long long pow_128(__int128_t n, long long k, long long mo) {
2     __int128_t ret = 1;
3     while (k) {
4         if (k & 1) (ret *= n) %= mo;
5         (n *= n) %= mo;
6         k >>= 1;
7     } return ret;
8 }
9
10 bool miller_rabin(long long n) {
11     static const long long jp[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
12
13     if (n <= 1 || (n > 2 && !(n & 1))) return false;
14     for (long long p : jp) if (n % p == 0) return n == p;
15
16     long long r = n - 1, x, y;
17     long long e = 0;
18     while (~r & 1) r >>= 1, ++e;
19     for (long long p : jp) {
20         x = pow_128(p, r, n);
21         for (long long t = 0; t < e && x > 1; ++t) {
22             y = (__int128_t)x * x % n;
23             if (y == 1 && x != n - 1) return false;
24             x = y;
25         }
26         if (x != 1) return false;
27     }
28     return true;
29 }
30

```